Supplementary
RayGaussX: Accelerating Gaussian-Based Ray Marching for Real-Time and High-Quality Novel View Synthesis

Supplementary Overview

This supplementary material is organized as follows. Section 1 discusses current limitations and outlines future work. Section 2 derives the upper bound on the volume ratio (see Sec. 3.4 of the main paper). We first outline a general derivation of the bound in Subsection 2.1, then compute the minimum AABB of an ellipsoid in Subsection 2.2, and finally establish an upper bound on its volume in Subsection 2.3. Section 3 connects the densification criterion with the projected-mean derivative. Section 4 experimentally demonstrates that the new densification criterion more effectively densifies regions far from camera poses. Section 5 presents a detailed ablation study evaluating the impact of each contribution. Section 6 presents a targeted comparison of our approach with two related methods: 3D Gaussian Splatting (3D-GS) [9] and 3D Gaussian Ray Tracing (3DGRT) [16]. First, we compare rendering quality using the same appearance model (spherical harmonics / spherical Gaussians) for both 3D-GS and our method to isolate the impact of their rendering pipelines. Second, we analyze the different constraints of our approach and 3DGRT that motivate our choice of enclosing volumes. Section 7 presents additional experimental results in two parts: Subsection 7.1 shows qualitative comparisons, and Subsection 7.2 provides detailed quantitative evaluations.

1. Limitations and Future Work

The proposed approach achieves state-of-the-art rendering quality with a reasonable training time of 60–80 minutes and enables interactive rendering. However, this requires a high-end GPU (NVIDIA RTX 4090 in our experiments), whereas Gaussian Splatting and its variants can run on mobile devices or in web-GL environments. Future work could further accelerate rendering with novel optimizations. Additionally, our approach does not properly handle aliasing, which falls outside the scope of this paper and could be addressed by future work. Nevertheless, RayGaussX's fast training and high-quality rendering make it a strong framework for applications requiring high accuracy, including surface reconstruction [6], inverse rendering [11], SLAM [14], camera optimization [12], and relighting [3].

2. Computation of the upper bound on the ratio of volumes

In this section, we derive the upper bound used for the volume ratio in **Section 3.4 Limiting Highly Anisotropic Gaussian** of the main paper. First, in Subsection 2.1, we present a general derivation of this bound by assuming intermediate results, which are then proven in Subsections 2.2 and 2.3.

2.1. Derivation of the Volume Ratio Bound

We consider here the computation of an upper bound for the ratio:

$$r = \frac{\mathrm{Vol}(\mathrm{AABB}(\mathcal{E}))}{\mathrm{Vol}(\mathcal{E})} \tag{1}$$

with Vol the volume function, $\mathcal{E}$ a given ellipsoid, and $\mathrm{AABB}(\mathcal{E})$ its associated axis-aligned bounding box.

In particular, if we consider the l-th primitive of our representation, the ellipsoid associated with the l-th Gaussian is the σ_ϵ-level isosurface of the density function of the l-th primitive:

$$\sigma_l(\mathbf{x}) = \tilde{\sigma_l} \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{l})^T {\mathbf{\Sigma_l}}^{-1}(\mathbf{x}-\mu_\mathbf{l})\right) \tag{2}$$

where σ_ϵ is the chosen density threshold [4]. In order to avoid overloading the notations, we omit the index l hereafter. The covariance matrix can be decomposed as: $\mathbf{\Sigma} = \mathbf{R}\mathbf{S}\mathbf{S}^T\mathbf{R}^T$ where $\mathbf{R} = (r_{i,j})_{1\le i,j\le 3}$ is a rotation matrix and $\mathbf{S} = \mathrm{diag}(s_1, s_2, s_3)$ a scale matrix. Hence, the ellipsoid associated with the primitive can be described by the equation:

$$(\mathbf{x}-\mu)^T\tilde{\mathbf{\Sigma}}^{-1}(\mathbf{x}-\mu) \le 1 \tag{3}$$

such that $\tilde{\mathbf{\Sigma}} = \mathbf{R}\,\tilde{\mathbf{S}}\,\tilde{\mathbf{S}}^T\mathbf{R}^T$, where $\tilde{\mathbf{S}} = \sqrt{2\,\ln\!\left(\frac{\tilde{\sigma}}{\sigma_\epsilon}\right)}\cdot\mathbf{S}$. The volume of ellipsoid $\mathcal{E}$ is then:

$$\mathrm{Vol}(\mathcal{E}) = \frac{4\pi}{3}\left[2\ln\!\left(\frac{\tilde{\sigma}}{\sigma_\epsilon}\right)\right]^{3/2} s_1\, s_2\, s_3$$

Also, it can be shown that the volume of the associated AABB, Vol(AABB($\mathcal{E}$)), is equal to (see Subsection 2.2 for details):

$$8\left(2\,\ln\!\left(\frac{\tilde{\sigma}}{\sigma_\epsilon}\right)\right)^{\frac{3}{2}}\sqrt{\prod_{i=1}^{3}\left(r_{i,1}^2\,s_1^2+r_{i,2}^2 s_2^2+r_{i,3}^2 s_3^2\right)} \tag{4}$$

And therefore, the ratio of the volumes is:

$$r=\frac{6}{\pi}\,\frac{\sqrt{\prod_{i=1}^{3}(r_{i,1}^2\,s_1^2+r_{i,2}^2\,s_2^2+r_{i,3}^2\,s_3^2)}}{s_1\,s_2\,s_3} \tag{5}$$

independent of the scale factor introduced by σ_ϵ and $\tilde{\sigma}$. Subsequently, we ensure that this ratio remains rotation-invariant to avoid bias toward any particular Gaussian orientation, and we constrain the worst-case scenario by computing an upper bound (see Subsection 2.3 for details):

$$\frac{\mathrm{Vol}(\mathrm{AABB}(\mathcal{E}))}{\mathrm{Vol}(\mathcal{E})}\leq\frac{2}{\pi\sqrt{3}}\,\frac{\left(s_1^2+s_2^2+s_3^2\right)^{\frac{3}{2}}}{s_1\,s_2\,s_3} \tag{6}$$

Thus, we obtain the expression for the upper bound used in the main article in the section **Limiting Highly Anisotropic Gaussian**.

2.2. Computation of the Minimum Axis-Aligned Bounding Box for an Ellipsoid

Considering the ellipsoid $\mathcal{E}$ defined in global coordinates by

$$(x-\mu)^T\tilde{\Sigma}^{-1}(x-\mu)\leq 1 \tag{7}$$

with: $\tilde{\Sigma}=R\,\tilde{S}\,\tilde{S}^T R^T$ and $\tilde{S}=\mathrm{diag}(\tilde{s}_1,\tilde{s}_2,\tilde{s}_3)$, where the rotation matrix $R=(r_{ij})$ performs the transformation of the ellipsoid (aligned in its local coordinate system) into the global coordinate system.

In this section, we determine the minimum axis-aligned bounding box for the ellipsoid $\mathcal{E}$. To obtain the axis-aligned bounding box in the world reference frame, we need to determine, for each axis i in the world frame (with $i=1,2,3$ corresponding to x,y, and z), the maximum half-length L_i such that

$$L_i=\max_{x\in\mathbb{R}^3}\left\{|x_i-\mu_i|\;:\;(x-\mu)^T\tilde{\Sigma}^{-1}(x-\mu)\leq 1\right\} \tag{8}$$

The volume of the axis-aligned bounding box (AABB) will then be computed as

$$\mathrm{Vol}(\mathrm{AABB}(\mathcal{E}))=\prod_{i=1}^{3}2L_i \tag{9}$$

First, we introduce the local variable in the ellipsoid-centered reference frame: $y=R^T(x-\mu)$ and thus we obtain: $(x-\mu)=Ry$ thanks to the orthogonality of R. Similarly, the half-lengths can be re-expressed using the local coordinate y in the ellipsoid's reference frame, using the fact that:

$$|x_i-\mu_i|=|[x-\mu]_i|=\left|\sum_{j=1}^{3}r_{ij}y_j\right| \tag{10}$$

and:

$$(x-\mu)^T\tilde{\Sigma}^{-1}(x-\mu)=y^T(\tilde{S}\tilde{S}^T)^{-1}y\leq 1 \tag{11}$$

Since $\tilde{S}$ is diagonal, this constraint can be rewritten as:

$$\sum_{j=1}^{3}\left(\frac{y_j}{\tilde{s}_j}\right)^2\leq 1$$

Thus, we obtain the new expression for the half-lengths:

$$L_i=\max_{y\in\mathbb{R}^3}\left\{\left|\sum_{j=1}^{3}r_{ij}y_j\right|\;:\;\sum_{j=1}^{3}\left(\frac{y_j}{\tilde{s}_j}\right)^2\leq 1\right\}. \tag{12}$$

We now introduce the variable $z_j=\frac{y_j}{\tilde{s}_j}$ and reformulate the problem in terms of the variable z, which belongs to the unit ball in $\mathbb{R}^3$, as follows:

$$L_i=\max_{z\in\mathbb{R}^3}\left\{\left|\sum_{j=1}^{3}r_{ij}\tilde{s}_j z_j\right|\;:\;\sum_{j=1}^{3}(z_j)^2\leq 1\right\}. \tag{13}$$

Here, using the Cauchy-Schwarz inequality, we obtain:

$$L_i=\sqrt{\sum_{j=1}^{3}r_{ij}^2\tilde{s}_j^2} \tag{14}$$

where the maximum is attained for the vector collinear to $\mathbf{r}_{i,\cdot}\tilde{\mathbf{s}}$ with unit norm. Consequently, the volume of the axis-aligned bounding box can be computed as:

$$\mathrm{Vol}(\mathrm{AABB}(\mathcal{E}))=8\sqrt{\prod_{i=1}^{3}\sum_{j=1}^{3}r_{ij}^2\tilde{s}_j^2} \tag{15}$$

Thus, we derive the formula used to compute the expression of r in Section 2.

2.3. Upper Bound on the Volume of the Enclosing Axis-Aligned Bounding Box

We express the volume of the AABB enclosing the ellipsoid in the form:

$$\mathrm{Vol}(\mathrm{AABB}(\mathcal{E}))=8\sqrt{\prod_{i=1}^{3}\sum_{j=1}^{3}r_{ij}^2 s_j^2} \tag{16}$$

We use s instead of $\tilde{s}$ because, at the stage of the section 2 requiring this upper bound, the scaling accounting for σ and σ_ϵ is no longer necessary.

Here, we apply the arithmetic-geometric inequality to $l_i = \sum_{j=1}^{3} r_{ij}^2 s_j^2$, yielding:

$$l_1 l_2 l_3 \leq \left(\frac{l_1 + l_2 + l_3}{3} \right)^3 \tag{17}$$

with equality when $l_1 = l_2 = l_3$. Then, we have:

$$\sum_{i=1}^{3} l_i = \sum_{i=1}^{3} \sum_{j=1}^{3} r_{ij}^2 s_j^2 \tag{18}$$

We can note here that, since the matrix R is orthogonal, its columns form an orthonormal basis, thus $\sum_{i=1}^{3} r_{ij}^2 = 1$, yielding:

$$\sum_{i=1}^{3} l_i = \sum_{j=1}^{3} s_j^2 \tag{19}$$

and:

$$l_1 l_2 l_3 \leq \left(\frac{\sum_{j=1}^{3} s_j^2}{3} \right)^3 \tag{20}$$

Thus, knowing that $\mathrm{Vol}(\mathrm{AABB}(\mathcal{E})) = 8\sqrt{l_1 l_2 l_3}$, and by using the equations (20), we obtain:

$$\mathrm{Vol}(\mathrm{AABB}(\mathcal{E})) \leq \frac{8}{3\sqrt{3}} \left(\sum_{j=1}^{3} s_j^2 \right)^{\frac{3}{2}} \tag{21}$$

This expression enables us to establish an upper bound on the ratio r in Section 2. Moreover, this upper bound is achieved when the half-lengths are equal $L_1 = L_2 = L_3$ (since $L_i = \sqrt{l_i}$).

3. Connecting the Densification Criterion and the Projected Mean Derivative

In this section, o denotes the center of the camera, f its focal distance, and μ the position of a Gaussian. We will see here that the densification criterion can be expressed in terms of the gradient with respect to the position μ projected onto the sphere centered at o with radius f, under well-chosen assumptions. We define the projection $P : \mathbb{R}^3 \setminus \{o\} \to \mathbb{R}^3$ by:

$$P(\mu) = o + f \, \frac{\mu - o}{\|\mu - o\|} \tag{22}$$

where $f > 0$ is the camera's focal distance and o is the camera center. Thus, $P(\mu)$ is always a point on the sphere of radius f centered at o. Here, we introduce a new parameterization based on the position on the sphere. We define $\tilde{\mu} = (\mu_P, r)$ where:

- $\mu_P = P(\mu) = o + f \, \frac{\mu - o}{\|\mu - o\|}$ is the projection of $\mu \in \mathbb{R}^3 \setminus \{o\}$ onto the sphere $S^2(o, f)$ (the sphere of radius $f > 0$ centered at o).
- $r = \|\mu - o\|$ is the radial distance of μ from o.

We then define the reprojection function $F : S^2(o, f) \times \mathbb{R}_+ \to \mathbb{R}^3$ such that:

$$\mu = F(\mu_P, r) = o + \frac{r}{f} (\mu_P - o) \tag{23}$$

Also, by introducing the unit vector $u = \frac{\mu - o}{\|\mu - o\|}$, we note that the derivatives of F with respect to the parameters can be expressed as follows:

$$\frac{\partial F}{\partial \mu_P} = \frac{r}{f} \left(I - u u^T \right) \tag{24}$$

where $I - uu^T$ is the projector onto $T_{\mu_P} S^2(o, f) = \{v \in \mathbb{R}^3 \mid v \cdot u = 0\}$, the tangent space to the sphere centered at o and with radius f at the point $\mu_P = o + fu$. This multiplication by the projection is necessary because μ_P is constrained to lie on the sphere $S^2(o, f)$; hence, its derivatives must lie in the tangent space $T_{\mu_P} S^2(o, f)$. Furthermore, we have:

$$\frac{\partial F}{\partial r} = \frac{1}{f} \left(\mu_P - o \right) \tag{25}$$

Thus, we obtain the gradients, with respect to μ_P:

$$\nabla_{\mu_P} L = \frac{r}{f} (I - uu^T) \nabla_\mu L \tag{26}$$

and the radial component:

$$\nabla_r L = \frac{1}{f} \left(\mu_P - o \right)^T \nabla_\mu L \tag{27}$$

Experimentally, we observe that the derivative along the radial direction is significantly smaller compared to the tangential components. Also, we assume in the following analysis that $\nabla_\mu L \cdot u = 0$. Thus, we have:

$$\nabla_\mu L \in T_{\mu_P} S^2(o, f) \tag{28}$$

Moreover, since $(I - uu^T)$ is a projector onto $T_{\mu_P} S^2(o, f)$ and $\nabla_\mu L$ lies in this tangent space, we have:

$$(I - uu^T) \nabla_\mu L = \nabla_\mu L. \tag{29}$$

and:

$$\left(\mu_P - o \right)^T \nabla_\mu L = 0 \tag{30}$$

Thus, we finally obtain that:

$$\nabla_{\mu_P} L = \frac{r}{f} \nabla_\mu L \tag{31}$$

and:

$$\nabla_r L = 0 \tag{32}$$

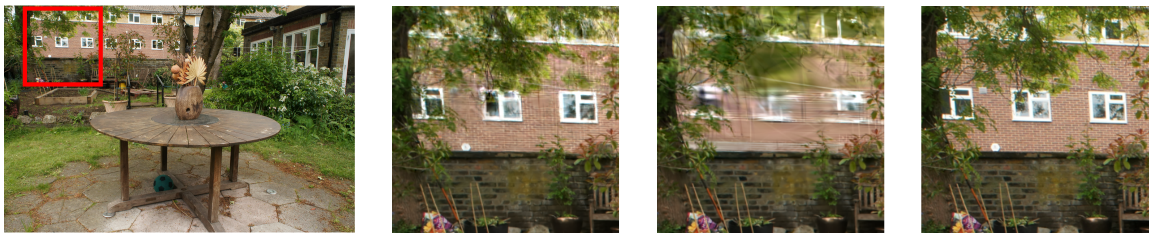


Figure 1. Comparison of background reconstruction for image 1 in the *garden* test scene. From left to right, the full scene with the region of interest highlighted in red, the reconstruction obtained with the new densification criterion, the reconstruction obtained with the previous densification criterion, and the corresponding ground-truth image crop.

Therefore, the gradient with respect to the projection on the sphere is, in this case, proportional to the gradient in the 3D space, and by taking the norm, we obtain:

$$\|\nabla_{\mu_P} L\| = \frac{r}{f}\,\|\nabla_{\mu} L\| \tag{33}$$

4. Experimental Evaluation of Densification

In this section, we experimentally demonstrate the effectiveness of the new densification criterion. To this end, we train the *garden* scene from the MipNeRF360 dataset first using the original criterion and then using the new densification criterion, with densification parameters tuned to yield approximately the same number of Gaussians in the scene (around 4 million). Subsequently, by considering the point clouds associated with the centers of the Gaussians, we compute the number of neighbors of each Gaussian within a radius of $R = 0.125$ and use this metric to denote the density of Gaussians around a given Gaussian. The point clouds associated with this scalar field are shown in Fig. 2. In this scene, the cameras are predominantly positioned around the central table. Under the new densification criterion, there remains a significant number of Gaussians around the table, which corresponds to the area with the most information, and there are also more points distributed in the distant regions. In particular, the right-hand side of the *garden* scene contains a higher density of Gaussians, confirming the effectiveness of the new criterion in better densifying distant areas.

Moreover, we demonstrate the effect of the new densification on rendering in Fig. 1 by comparing the background reconstruction of image 1 in the *garden* scene test set. As shown, the previous densification produces a poorly reconstructed background that appears blurry and lacks detail. In contrast, our new approach yields a much more accurate background reconstruction, with fine details such as the reflection on the glass pane clearly visible.

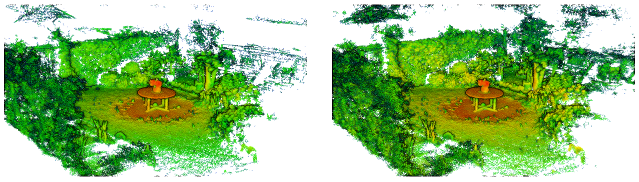

(a) Raygauss densification (b) RaygaussX densification

Figure 2. Densification comparison in the MipNeRF360 *garden* scene using Raygauss and RaygaussX criteria. The color gradient from blue through green, yellow, and red encodes on a logarithmic scale the increasing number of Gaussian neighbors within a radius of $R = 0.125$.

5. Detailed ablation study of each contribution

In this section, we present a detailed ablation study of the various contributions introduced in RayGaussX to complement the study presented in the main paper by detailing the influence of each component individually. Specifically, we investigate the influence of the main contributions: the novel densification strategy (D) which yields a more uniform distribution of Gaussians in regions distant from the camera poses, empty-space skipping (E), adaptive sampling (A), spatial reordering of Gaussians via a Z-order curve (Z), ray coherence optimization (R) and the isotropic loss (L).

This ablation study, presented in Table 1, is designed to determine how each individual contribution and their combinations influence rendering quality (PSNR, SSIM, LPIPS), training time, and inference speed in frames per second (FPS). First, rows (1) and (1*) demonstrate that implementation-level optimizations provide a stronger baseline than RayGauss in terms of training time and FPS, while preserving comparable rendering performance. These improvements are not described in the paper as they consist of using faster operations and pertain to implementation details. However, interested readers may compare our code, available online, with the implementation from the RayGauss paper.

Next, rows (2), (3), (5), (6), and (9) confirm that the core contributions of the paper effectively reduce both training

	D	E	A	Z	R	L	PSNR↑	SSIM↑	LPIPS↓	Train↓	FPS↑
(1)							28.06	0.879	0.103	585 min	0.5
(1*)							28.14	0.885	0.090	297 min	1.5
(2)		✓					28.14	0.884	0.090	218 min	4.4
(3)			✓				28.13	0.885	0.089	191 min	4.7
(4)		✓	✓				28.14	0.884	0.090	180 min	5.9
(5)				✓			28.14	0.885	0.089	237 min	2.4
(6)					✓		28.12	0.884	0.090	238 min	2.4
(7)				✓	✓		28.15	0.885	0.089	200 min	3.6
(8)		✓	✓	✓	✓		28.15	0.885	0.089	125 min	10.2
(9)						✓	28.12	0.885	0.090	133 min	10.1
(10)		✓	✓	✓	✓	✓	28.16	0.885	0.090	88 min	26.1
(11)	✓						28.38	0.887	0.090	294 min	1.8
(12)	✓	✓	✓				28.35	0.887	0.089	168 min	6.3
(13)	✓	✓	✓	✓	✓		28.36	0.888	0.089	116 min	10.5
(14)	✓	✓	✓	✓	✓	✓	28.35	0.887	0.090	84 min	27.4

Table 1. Ablation study of key contributions on the *garden* scene from the Mip-NeRF360 dataset (D: new Densification criterion; E: Empty-space Skipping; A: Adaptive sampling; Z: Z-curve re-indexing of Gaussians; R: Ray coherence; L: Loss for isotropic Gaussians). Row (1) represents the original RayGauss method [4], Row (1*) corresponds to the optimized RayGauss baseline underlying RayGaussX, incorporating implementation-level optimizations but excluding the main contributions (D, E, A, Z, R, L) and row (14) corresponds to our full method, RayGaussX.

and inference times. We also observe that overall rendering quality is minimally affected by these contributions, demonstrating that they accelerate rendering without any significant trade-off in quality. The isotropic loss appears to slightly reduce PSNR, but this effect is minor.

Rows (4), (7), (8), and (10) illustrate the cumulative effect of the individual contributions. We observe no accumulation of errors that could degrade the rendering-quality metrics; indeed, the PSNR remains approximately 28.14 dB. This confirms that, with the chosen hyperparameters, the contributions do not compromise rendering quality. Furthermore, training and inference are accelerated when combining these contributions, achieving a training time of 88 minutes and an inference speed of 26.1 FPS when all speed-related contributions are applied.

Finally, rows (11), (12), (13), and (14) highlight the influence of the new densification on the algorithm's performance. It is clear that the new densification yields superior rendering quality by better densifying distant regions, as shown in Section 4. In particular, with the new densification we achieve a PSNR of approximately 28.36 dB compared to 28.14 dB obtained with (1*). Furthermore, by comparing rows (12), (13), and (14) to their counterparts without the new densification, we observe that the new densification also appears to slightly accelerate rendering and inference times.

6. Selective comparison with related methods

In this section, we present a targeted comparison of our approach with two related methods: 3D Gaussian Splatting (3D-GS) [9] and 3D Gaussian Ray Tracing (3DGRT) [16]. First, we compare rendering quality using the same appearance model, spherical harmonics/spherical Gaussians, for both 3D-GS and our method to isolate the impact of their rendering pipelines. Second, we analyze the different constraints of our approach and 3DGRT that motivate our choice of enclosing volumes.

6.1. Rendering quality comparison with a similar appearance model

In this subsection, we compare RayGaussX with 3D Gaussian Splatting (3D-GS) under the same appearance model (spherical harmonics and spherical Gaussians). Using an identical appearance representation isolates the rendering algorithm's impact on output quality, since both methods employ Gaussian primitives within an identical appearance framework. We evaluate both methods on the datasets introduced in the main paper: NeRF-Synthetic,

Method	NeRF-Synth.	NSVF-Synth.	Mip-NeRF360	Tanks & Temp.	Deep Blending
3D-GS	33.39/0.968	37.07/0.987	27.80/0.825	23.72/0.848	29.92/0.905
3D-GS (SH, SG)	33.88/0.971	38.06/0.988	28.12/0.827	**23.81**/0.855	29.88/0.908
RayGaussX (ours)	**34.54/0.974**	**38.75/0.991**	**28.43/0.842**	23.76/**0.865**	**30.32/0.915**

Table 2. Comparison in terms of PSNR/SSIM of RayGaussX (ours) and 3D-GS methods (with and without spherical harmonics/gaussians) on NeRF-Synthetic, NSVF-Synthetic, Mip-NeRF360, Tanks and Temples, and Deep Blending datasets. Best results in bold.

NSVF-Synthetic, Mip-NeRF360, Tanks&Temples, and Deep Blending. The appearance model comprises nine spherical harmonics and seven spherical Gaussians per primitive. The resulting PSNR and SSIM values are reported in Fig. 2. Results show that the SH/SG formulation also enhances 3D-GS's performance. However, except for PSNR on Tanks&Temples, across all datasets and for every standard metric, RayGaussX outperforms the modified 3D-GS in terms of rendering quality, confirming the advantage of employing Volume Ray Marching. This aligns with theory, as ray marching provides a less approximate formulation that avoids rendering artifacts such as flickering compared to 3D-GS rasterization. Additionally, as noted by the authors of [1], the Tanks & Temples [10] dataset exhibits significant lighting variations, making conclusions drawn from this dataset somewhat uncertain.

6.2. Enclosing volume selection

The recent 3DGRT method [16] also uses the OptiX API to ray trace Gaussian primitives. However, their choice of enclosing volume differs from ours: they use bounding polyhedra and we use axis aligned bounding boxes(AABB). We explain the reasons for this difference in this subsection.

The 3D Gaussian Ray Tracing approach performs ray tracing of Gaussians with a single sample per Gaussian; this simplified approach approximates the rendering equation and, unlike RayGauss, does not account for the overlap of multiple Gaussians. Additionally, compared with AABBs, 3DGRT's complex bounding primitives raise two issues. First, constructing them and building the associated BVH are several orders of magnitude slower (Fig. 9 in 3DGRT paper), so the frequent BVH rebuilds required during training become a bottleneck in scenes with many primitives. A second drawback stems from using polyhedra with the OptiX API for ray marching: indeed, we integrate each ray segment sequentially. However, if a segment lies entirely within a bounding polyhedron, OptiX reports no intersection, wrongly treating it as empty because it crosses no faces. In contrast, with AABBs, an OptiX-specific behavior is that it reports an intersection even for fully enclosed segments. For these reasons, AABBs are better suited to our Volume Ray Marching algorithm than the bounding polyhedra proposed by 3DGRT.

7. Additional experimental results

7.1. Qualitative results

Fig. 3 presents qualitative results of our RayGaussX approach compared to RayGauss [4], Zip-NeRF [2], and 3D-GS [9] on the Mip-NeRF360 [1] dataset. First, we observe that the qualitative results are very similar between RayGaussX and RayGauss for indoor scenes. However, for outdoor scenes, our approach shows a clear improvement in background rendering thanks to our new densification criterion. The Zip-NeRF method achieves very good visual results but produces grainy images (particularly noticeable in the *counter* scene) and sometimes generates floaters, as seen on the wooden foot in the *treehill* scene. Finally, 3D-GS exhibits lower reconstruction quality, with less accurately reconstructed objects, which is especially visible in the shelf of the *room* scene.

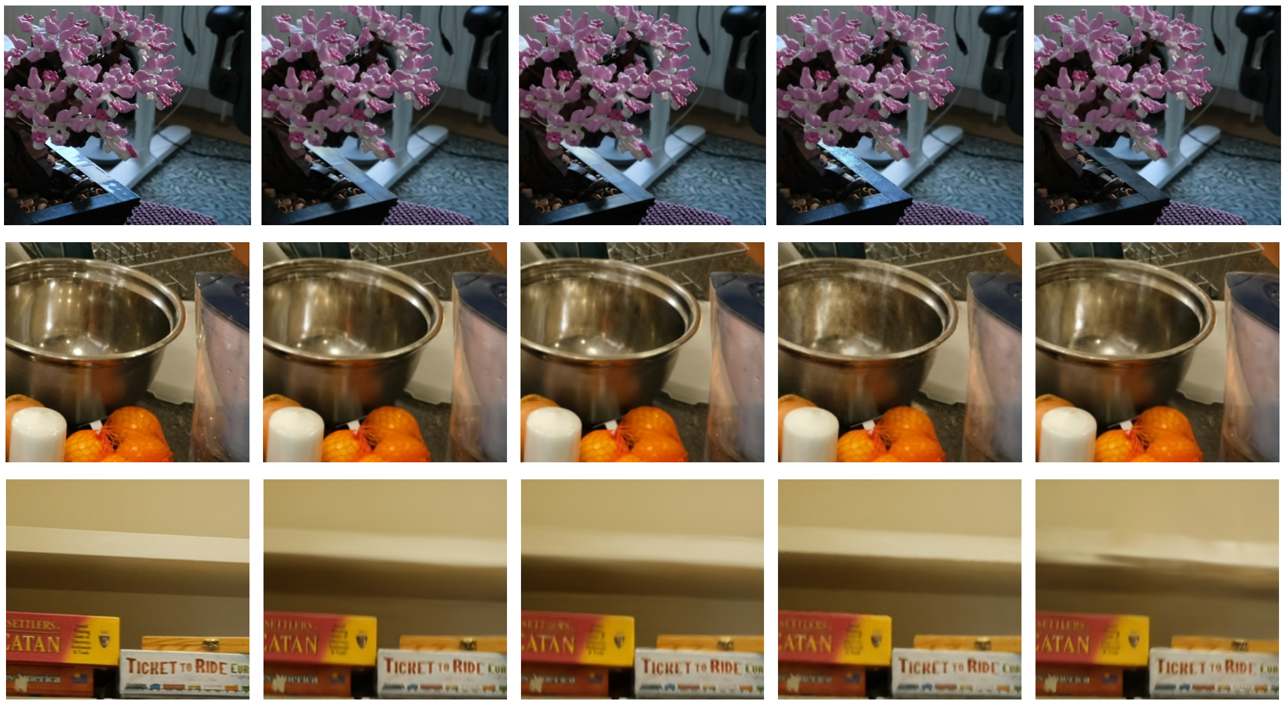
As noted by the authors of [1], the Tanks&Temples [10] dataset contains images with significant variations in lighting conditions, making conclusions drawn from this dataset somewhat uncertain. Moreover, we can observe that the Zip-NeRF [2] method failed on this dataset.

7.2. Detailed quantitative results

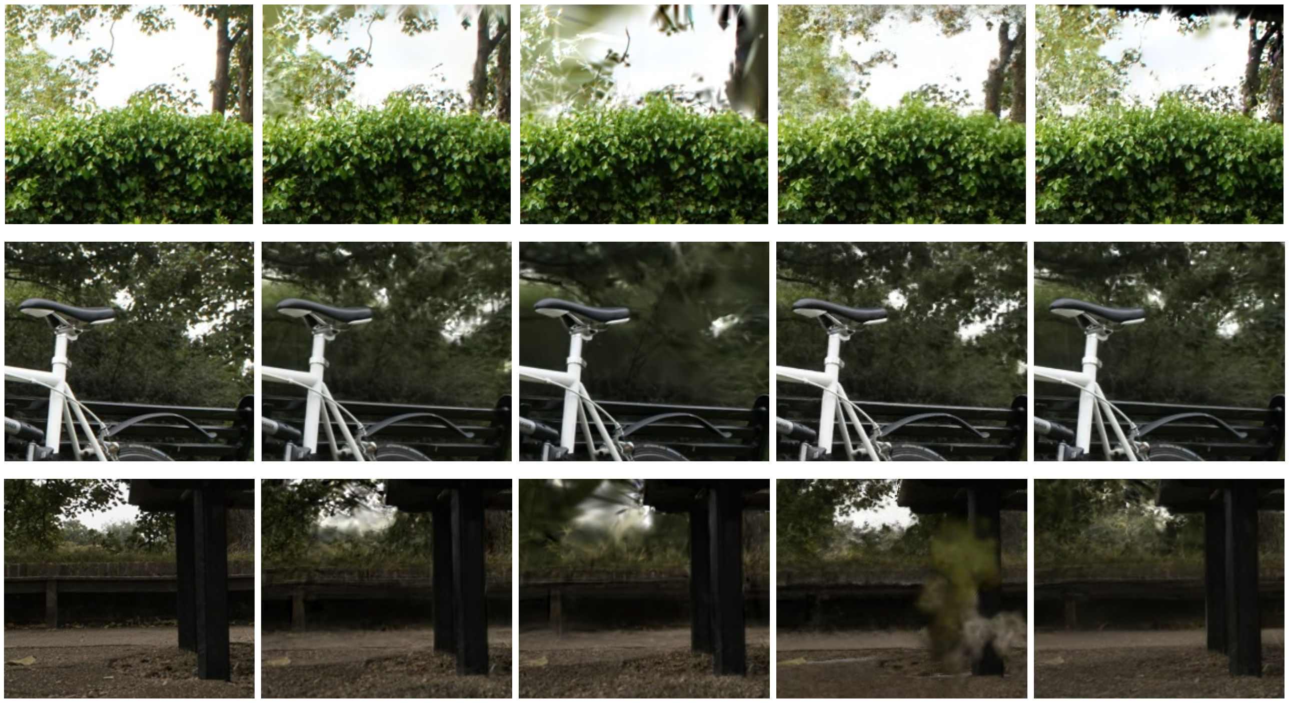
Tab. 3, Tab. 4, Tab. 5, and Tab. 6 show the detailed per scene results of the main paper with metrics PSNR, SSIM, and LPIPS on the NeRF Synthetic [15], NSVF Synthetic [13], Mip-NeRF360 [1], Tanks&Temples [10] and Deep Blending [8] datasets.

All methods with an * in tables of the main paper and supplementary have been re-trained using the available online code of respective methods:

- 3D Gaussian Splatting: https://github.com/graphdeco-inria/gaussian-splatting
- Mip-Splatting: https://github.com/autonomousvision/mip-splatting
- Spec-Gaussian: https://github.com/ingra14m/Specular-Gaussians
- Zip-NeRF: https://github.com/jonbarron/camp_zipnerf
- RayGauss: https://github.com/hugobl1/ray_gauss



(a) Ground Truth (b) RayGaussX (ours) (c) RayGauss [4] (d) Zip-NeRF [2] (e) 3D-GS [9]



(f) Ground Truth (g) RayGaussX (ours) (h) RayGauss [4] (i) Zip-NeRF [2] (j) 3D-GS [9]

Figure 3. Qualitative results on the Mip-NeRF360 [1] dataset. The scenes are from the top down: *bonsai*, *counter*, *room*, *garden*, *bicycle*, *treehill*.

	PSNR ↑								
	chair	drums	ficus	hotdog	lego	materials	mic	ship	**Avg.**
Instant-NGP [17]	35.00	26.02	33.51	37.40	36.39	29.78	36.22	31.10	33.18
Mip-NeRF360 [1]	35.65	25.60	33.19	37.71	36.10	29.90	36.52	31.26	33.24
Point-NeRF [18]	35.40	26.06	36.13	37.30	35.04	29.61	35.95	30.97	33.30
3D-GS [9]*	35.85	26.22	35.00	37.81	35.87	30.00	35.40	30.95	33.39
Zip-NeRF [2]*	35.78	25.91	34.72	38.05	35.79	31.05	35.92	32.33	33.69
Mip-Splatting [20]*	36.23	26.29	35.33	38.04	36.13	30.37	36.22	31.32	33.74
Spec-Gaussian [19]*	36.20	26.76	35.56	38.04	36.06	30.61	35.78	31.45	33.80
3DGRT [16]*	35.69	25.88	36.54	37.98	36.80	30.40	35.88	31.73	33.86
NeuRBF [7]*	36.54	26.38	35.01	38.44	37.35	34.12	36.16	31.73	34.47
RayGauss [4]*	37.20	27.14	35.11	38.30	37.10	31.36	38.11	31.95	34.53
RayGaussX (ours)	37.19	27.10	35.09	38.46	37.01	31.33	38.02	32.13	34.54

	SSIM ↑								
	chair	drums	ficus	hotdog	lego	materials	mic	ship	**Avg.**
Instant-NGP [17]	0.979	0.937	0.981	0.982	0.982	0.951	0.990	0.896	0.963
Mip-NeRF360 [1]	0.983	0.931	0.979	0.982	0.980	0.949	0.991	0.893	0.961
Point-NeRF [18]	0.984	0.935	0.987	0.982	0.978	0.948	0.990	0.892	0.962
3D-GS [9]*	0.988	0.955	0.988	0.986	0.983	0.960	0.992	0.893	0.968
Zip-NeRF [2]*	0.987	0.948	0.987	0.987	0.983	0.968	0.992	0.937	0.974
Mip-Splatting [20]*	0.988	0.955	0.988	0.987	0.984	0.963	0.993	0.908	0.971
Spec-Gaussian [19]*	0.987	0.955	0.988	0.986	0.983	0.964	0.992	0.906	0.970
3DGRT [16]*	0.987	0.954	0.989	0.986	0.985	0.961	0.991	0.909	0.970
NeuRBF[7]*	0.988	0.944	0.987	0.987	0.986	0.979	0.992	0.925	0.974
RayGauss [4]*	0.990	0.960	0.988	0.988	0.986	0.969	0.995	0.914	0.974
RayGaussX (ours)	0.990	0.959	0.988	0.988	0.986	0.969	0.995	0.914	0.974

	LPIPS ↓								
	chair	drums	ficus	hotdog	lego	materials	mic	ship	**Avg.**
Instant-NGP [17]	0.022	0.071	0.023	0.027	0.017	0.060	0.010	0.132	0.045
Mip-NeRF360 [1]	0.018	0.069	0.022	0.024	0.018	0.053	0.011	0.119	0.042
Point-NeRF [18]	0.023	0.078	0.022	0.037	0.024	0.072	0.014	0.124	0.049
3D-GS [9]*	0.011	0.037	0.011	0.017	0.015	0.034	0.006	0.118	0.031
Zip-NeRF [2]*	0.013	0.045	0.013	0.017	0.015	0.031	0.006	0.082	0.028
Mip-Splatting [20]*	0.013	0.038	0.011	0.017	0.015	0.033	0.006	0.098	0.029
Spec-Gaussian [19]*	0.011	0.034	0.011	0.017	0.015	0.030	0.005	0.094	0.027
3DGRT [16]*	0.016	0.047	0.013	0.024	0.016	0.046	0.009	0.123	0.037
NeuRBF[7]*	0.016	0.061	0.016	0.021	0.015	0.032	0.008	0.114	0.035
RayGauss [4]*	0.009	0.030	0.011	0.015	0.012	0.026	0.004	0.088	0.024
RayGaussX (ours)	0.009	0.030	0.010	0.015	0.012	0.026	0.004	0.088	0.024

Table 3. **PSNR, SSIM and LPIPS (with VGG network) scores on the NeRF Synthetic dataset [15].** All methods are trained on the train set with full-resolution images (800x800 pixels) and evaluated on the test set with full-resolution images (800x800 pixels). Methods marked * were retrained on an NVIDIA RTX4090.

	PSNR ↑								
	Bike	Life	Palace	Robot	Space	Steam	Toad	Wine	**Avg.**
TensoRF [5]	39.23	34.51	37.56	38.26	38.60	37.87	34.85	31.32	36.53
3D-GS [9]*	40.76	33.19	38.89	39.16	36.80	37.67	37.33	32.76	37.07
Mip-Splatting [20]*	40.34	35.06	39.00	39.34	37.11	38.42	36.64	32.30	37.28
NeuRBF [7]	40.71	36.08	38.93	39.13	40.44	38.35	35.73	32.99	37.80
Spec-Gaussian [19]*	40.93	36.17	39.39	39.64	40.12	38.08	36.63	32.82	37.97
RayGauss [4]*	41.25	36.21	40.30	40.37	40.03	39.08	38.39	34.12	38.72
RayGaussX (ours)	41.34	36.39	40.28	40.32	40.11	39.12	38.39	34.01	38.75

	SSIM ↑								
	Bike	Life	Palace	Robot	Space	Steam	Toad	Wine	**Avg.**
TensoRF [5]	0.993	0.968	0.979	0.994	0.989	0.991	0.978	0.961	0.982
3D-GS [9]*	0.994	0.979	0.983	0.994	0.991	0.993	0.985	0.975	0.987
Mip-Splatting [20]*	0.995	0.982	0.985	0.996	0.992	0.994	0.985	0.978	0.988
NeuRBF[7]	0.995	0.977	0.985	0.995	0.993	0.993	0.983	0.972	0.987
Spec-Gaussian [19]*	0.995	0.982	0.986	0.996	0.994	0.994	0.985	0.977	0.989
RayGauss [4]*	0.996	0.983	0.987	0.996	0.994	0.994	0.989	0.981	0.990
RayGaussX (ours)	0.996	0.984	0.989	0.996	0.995	0.995	0.989	0.981	0.991

	LPIPS ↓								
	Bike	Life	Palace	Robot	Space	Steam	Toad	Wine	**Avg.**
TensoRF [5]	0.010	0.048	0.022	0.010	0.020	0.017	0.031	0.051	0.026
3D-GS [9]*	0.005	0.028	0.017	0.006	0.009	0.007	0.018	0.025	0.014
Mip-Splatting [20]*	0.005	0.024	0.015	0.007	0.010	0.008	0.019	0.022	0.014
NeuRBF[7]	0.006	0.036	0.016	0.009	0.011	0.011	0.025	0.036	0.019
Spec-Gaussian [19]*	0.004	0.022	0.013	0.006	0.007	0.007	0.016	0.020	0.012
RayGauss [4]*	0.003	0.022	0.011	0.006	0.007	0.006	0.012	0.017	0.011
RayGaussX (ours)	0.003	0.021	0.011	0.006	0.006	0.006	0.011	0.018	0.010

Table 4. **PSNR, SSIM and LPIPS (with VGG network) scores on the NSVF Synthetic dataset [13].** All methods are trained on the train set with full-resolution images (800x800 pixels) and evaluated on the test set with full-resolution images (800x800 pixels). Methods marked * were retrained on an NVIDIA RTX4090.

	PSNR ↑									
	bonsai	counter	kitchen	room	bicycle	flowers	garden	stump	treehill	**Avg.**
Instant-NGP [17]	30.69	26.69	29.48	29.69	22.17	20.65	25.07	23.47	22.37	25.59
3DGRT [16]*	31.93	28.51	29.67	30.68	24.74	21.36	26.82	26.25	22.05	26.89
Mip-NeRF360 [1]	33.46	29.55	32.23	31.63	24.37	21.73	26.98	26.40	22.87	27.69
3D-GS [9]*	32.62	29.17	31.39	31.96	25.65	21.77	27.62	27.00	23.00	27.80
Mip-Splatting [20]*	32.42	29.41	31.85	31.68	25.97	22.08	27.97	27.31	22.71	27.93
Spec-Gaussian [19]*	33.26	29.86	32.45	32.15	25.81	21.52	28.00	27.17	22.23	28.05
Zip-NeRF [2]*	34.79	29.12	32.36	33.04	25.85	22.33	28.22	27.35	23.95	28.56
RayGauss [4]*	33.91	30.56	32.83	31.83	25.51	21.85	28.06	26.33	23.18	28.23
RayGaussX (ours)	34.06	30.64	32.92	32.11	25.71	22.31	28.35	26.66	23.15	28.43

	SSIM ↑									
	bonsai	counter	kitchen	room	bicycle	flowers	garden	stump	treehill	**Avg.**
Instant-NGP [17]	0.906	0.817	0.858	0.871	0.512	0.486	0.701	0.594	0.542	0.699
3DGRT [16]*	0.941	0.904	0.914	0.914	0.744	0.612	0.848	0.767	0.621	0.807
Mip-NeRF360 [1]	0.941	0.894	0.920	0.913	0.685	0.583	0.813	0.744	0.632	0.792
3D-GS [9]*	0.947	0.916	0.933	0.929	0.777	0.618	0.868	0.783	0.654	0.825
Mip-Splatting [20]*	0.952	0.921	0.937	0.933	0.803	0.656	0.885	0.801	0.657	0.838
Spec-Gaussian [19]*	0.954	0.923	0.938	0.935	0.796	0.647	0.881	0.796	0.645	0.835
Zip-NeRF [2]*	0.952	0.905	0.929	0.929	0.772	0.637	0.863	0.788	0.674	0.828
RayGauss [4]*	0.957	0.932	0.942	0.936	0.782	0.634	0.879	0.775	0.672	0.834
RayGaussX (ours)	0.959	0.932	0.943	0.939	0.797	0.655	0.887	0.789	0.677	0.842

	LPIPS ↓									
	bonsai	counter	kitchen	room	bicycle	flowers	garden	stump	treehill	**Avg.**
Instant-NGP [17]	0.205	0.306	0.195	0.261	0.446	0.441	0.257	0.421	0.450	0.331
3DGRT [16]*	0.249	0.258	0.169	0.295	0.254	0.335	0.145	0.251	0.372	0.259
Mip-NeRF360 [1]	0.176	0.204	0.127	0.211	0.301	0.344	0.170	0.261	0.339	0.237
3D-GS [9]*	0.178	0.181	0.114	0.195	0.213	0.336	0.115	0.211	0.326	0.208
Mip-Splatting [20]*	0.161	0.167	0.109	0.177	0.164	0.266	0.090	0.182	0.270	0.176
Spec-Gaussian [19]*	0.163	0.167	0.108	0.178	0.167	0.263	0.093	0.185	0.270	0.177
Zip-NeRF [2]*	0.134	0.160	0.102	0.159	0.198	0.276	0.117	0.199	0.239	0.176
RayGauss [4]*	0.163	0.157	0.105	0.178	0.205	0.306	0.103	0.221	0.297	0.193
RayGaussX (ours)	0.154	0.156	0.103	0.171	0.177	0.285	0.090	0.200	0.262	0.177

Table 5. **PSNR, SSIM and LPIPS (with VGG network) scores on the Mip-NeRF360 dataset [15].** All methods are trained and tested on downsampled images by a factor of 2 in indoor (*bonsai*, *counter*, *kitchen*, *room*) and 4 in outdoor (*bicycle*, *flowers*, *garden*, *stump*, *treehill*). Methods marked * were retrained on an NVIDIA RTX4090.

	PSNR ↑					
	train	truck	**Avg.**	drjohnson	playroom	**Avg.**
Instant-NGP [17]	20.46	23.38	21.92	28.26	21.67	24.97
3DGRT [16]*	21.06	24.41	22.74	29.16	30.33	29.74
Mip-NeRF360 [1]	19.52	24.91	22.22	29.14	29.66	29.40
3D-GS [9]*	22.03	25.41	23.72	29.52	30.32	29.92
Mip-Splatting [20]*	21.78	25.66	23.72	28.83	30.19	29.51
Spec-Gaussian [19]*	22.10	25.62	23.86	29.03	30.31	29.67
Zip-NeRF [2]*	10.53	11.16	10.85	30.28	31.23	30.76
RayGauss [4]*	21.80	24.59	23.20	29.74	30.85	30.30
RayGaussX (ours)	22.24	25.27	23.76	29.85	30.79	30.32

	SSIM ↑					
	train	truck	**Avg.**	drjohnson	playroom	**Avg.**
Instant-NGP [17]	0.689	0.800	0.745	0.854	0.779	0.817
3DGRT [16]*	0.814	0.873	0.844	0.904	0.906	0.905
Mip-NeRF360 [1]	0.660	0.857	0.759	0.901	0.900	0.901
3D-GS [9]*	0.816	0.880	0.848	0.904	0.905	0.905
Mip-Splatting [20]*	0.827	0.893	0.860	0.899	0.907	0.903
Spec-Gaussian [19]*	0.823	0.888	0.856	0.902	0.910	0.906
Zip-NeRF [2]*	0.300	0.351	0.326	0.912	0.915	0.914
RayGauss [4]*	0.814	0.883	0.849	0.912	0.916	0.914
RayGaussX (ours)	0.834	0.896	0.865	0.913	0.916	0.915

	LPIPS ↓					
	train	truck	**Avg.**	drjohnson	playroom	**Avg.**
Instant-NGP [17]	0.360	0.249	0.305	0.352	0.428	0.390
3DGRT [16]*	0.223	0.170	0.197	0.316	0.313	0.315
Mip-NeRF360 [1]	0.354	0.159	0.257	0.237	0.252	0.245
3D-GS [9]*	0.205	0.150	0.178	0.241	0.246	0.244
Mip-Splatting [20]*	0.190	0.123	0.157	0.246	0.239	0.243
Spec-Gaussian [19]*	0.196	0.135	0.166	0.245	0.245	0.245
Zip-NeRF [2]*	0.658	0.664	0.661	0.217	0.201	0.209
RayGauss [4]*	0.201	0.132	0.167	0.238	0.245	0.242
RayGaussX (ours)	0.183	0.117	0.150	0.242	0.242	0.242

Table 6. **PSNR, SSIM and LPIPS (with VGG network) scores on the Tanks&Temples [10] and Deep Blending [8] datasets.** All methods are trained and tested on full images. Methods marked * were retrained on an NVIDIA RTX4090.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6, 7, 8, 10, 11

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 6, 7, 8, 10, 11

[3] Zoubin Bi, Yixin Zeng, Chong Zeng, Fan Pei, Xiang Feng, Kun Zhou, and Hongzhi Wu. Gs3: Efficient relighting with triple gaussian splatting. In *SIGGRAPH Asia 2024 Conference Papers*, 2024. 1

[4] Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. Raygauss: Volumetric gaussian-based ray casting for photorealistic novel view synthesis, 2024. 1, 5, 6, 7, 8, 9, 10, 11

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 9

[6] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–12, 2024. 1

[7] Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. Neurbf: A neural fields representation with adaptive radial basis functions. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 8, 9

[8] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 37(6), 2018. 6, 11

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4), 2023. 1, 5, 6, 7, 8, 9, 10, 11

[10] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 6, 11

[11] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1

[12] Changkun Liu, Shuai Chen, Yash Sanjay Bhalgat, Siyan HU, Ming Cheng, Zirui Wang, Victor Adrian Prisacariu, and Tristan Braud. GS-CPR: Efficient camera pose refinement via 3d gaussian splatting. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. 1

[13] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 6, 9

[14] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting slam. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1

[15] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. 6, 8, 10

[16] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics and SIGGRAPH Asia*, 2024. 1, 5, 6, 8, 10, 11

[17] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), 2022. 8, 10, 11

[18] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5428–5438, 2022. 8

[19] Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024. 8, 9, 10, 11

[20] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, 2024. 8, 9, 10, 11